```
import os, re, string, random, time, datetime, requests, sendgrid, random, flask
import ibm_db
from sendgrid.helpers.mail import *
from flask import Flask, request, render_template, flash, redirect, url_for, session
from werkzeug.utils import secure_filename
from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
from clarifai_grpc.grpc.api import service_pb2, resources_pb2, service_pb2_grpc
from clarifai_grpc.grpc.api.status import status_code_pb2


UPLOAD_FOLDER = 'static/uploads'
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])
# SENDGRID_API_KEY = "SG.HwfSJ6D4Tba6O-h7fL1JlA.z2_qdNI-
iXOhrhdzsx05PiEPj3bbNKXF_Rms0eRis4c"
SENDGRID_API_KEY = "SG.UZW-
7lxWS0K8eF5jNlmQog.JP0_eTLDnZjxuL1AJuWhUIIiQNBrCeq2yVai_ZtP3LM"
app = Flask(__name__)
app.secret_key = "vrkjnutrition"
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024


conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;Security=SSL;PROTO
COL=TCPIP;UID=pzt20234;PWD=r7CB0AmR1QtOHfR4;","","")
#;SSLServerCertificate=DigiCertGlobalRootCA.crt


YOUR_CLARIFAI_API_KEY = "af4bc9886c744e998ee0e20f104b1518"
YOUR_APPLICATION_ID = "test"
SAMPLE_URL =
"https://res.cloudinary.com/swiggy/image/upload/f_auto,q_auto,fl_lossy/nxmlubuz0b1qixa29gov"
metadata = (("authorization", f"Key {YOUR_CLARIFAI_API_KEY}"),)
channel = ClarifaiChannel.get_grpc_channel()
stub = service_pb2_grpc.V2Stub(channel)
```

```
# RAPIDAPI_KEY = "74e62205b6msha6b4e69e0088de5p12c619jsn1ed9cc5e0727"
RAPIDAPI_KEY = "5d7e4c1885mshe7780054ed873d4p13e49ajsn806c1791b2b0"


def allowed_file(filename):
  return '.' in filename and \
    filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


def sendMail(to, title, text):
  sg = sendgrid.SendGridAPIClient(api_key=SENDGRID_API_KEY)
  from_email = Email("953019106012@smartinternz.com")
  to_email = To(to)
  subject = title
  content = Content("text/plain", text)
  mail = Mail(from_email, to_email, subject, content)
  response = sg.client.mail.send.post(request_body=mail.get())
  print(response.status_code)
  print(response.body)
  print(response.headers)



@app.route("/forgot-pw", methods=["GET", "POST"])
def forgotpw():
  if flask.request.method == "POST":
    data = flask.request.form
    username=data['username']
    code = ''.join(random.choices(string.ascii_letters, k=6))

    sql= "SELECT * FROM users WHERE username=?"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,username)
    ibm_db.execute(stmt)
    account=ibm_db.fetch_assoc(stmt)
    print(account)
    session['userid'] = account['USERID']

    insert_sql = "INSERT INTO VERIFY VALUES(?,?)"
    prep_stmt=ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, account['USERID'])
    ibm_db.bind_param(prep_stmt, 2, code)
    ibm_db.execute(prep_stmt)

    sendMail(account['EMAIL'], "Verification Code", code)
    flash("We have sent a code to your registered email. please check spam folder also.")
    return redirect(url_for("confirmMail"))
  flash("We will send you a confirmation code to your registered email")
  return render_template("forgot-pw.html")


@app.route("/confirm-mail", methods=["GET", "POST"])
def confirmMail():
  session['LoggedIn'] = False
  if flask.request.method == "POST":
    data = flask.request.form
    usercode=data['code']

    sql= "SELECT * FROM verify WHERE userid=?"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['userid'])
    ibm_db.execute(stmt)
    verify=ibm_db.fetch_assoc(stmt)
    print(verify)
    dbcode = verify['CODE']
    if usercode == dbcode:
      session['LoggedIn'] = True
      delete_sql = "DELETE FROM verify WHERE CODE=?"
      prep_stmt=ibm_db.prepare(conn, delete_sql)
      ibm_db.bind_param(prep_stmt, 1, dbcode)
      ibm_db.execute(prep_stmt)
      flash("Email verified. Enter new password")
      return redirect(url_for("changepw"))
    else:
      flash("Error")
      return render_template("confirm-mail")
  return render_template("confirm-mail.html")


@app.route("/change-pw", methods=["GET", "POST"])
def changepw():
  if flask.request.method == "POST" and session['LoggedIn']:
    data = flask.request.form
    password=data['pw']
    sql = "UPDATE users SET PASSWORD=? WHERE USERID=?"
    prep_stmt=ibm_db.prepare(conn, sql)
    print(password, session['userid'])
    ibm_db.bind_param(prep_stmt, 1, password)
    ibm_db.bind_param(prep_stmt, 2, session['userid'])
    ibm_db.execute(prep_stmt)
    flash("Password changed.")
    return redirect(url_for("login"))
  else:
    flash("verification error")
    redirect(url_for("confirmMail"))
  return render_template("change-pw.html")
```

```
@app.route("/register", methods=["GET", "POST"])
def reg():
  if flask.request.method == "POST":

    data = flask.request.form
    email=data['email']
    username=data['username']
    password=data['pw']

    sql= "SELECT * FROM users WHERE username=?"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,username)
    ibm_db.execute(stmt)
    account=ibm_db.fetch_assoc(stmt)
    print(account)
    if account:
      flash("Account already exists!")
    elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
      flash("invalid email address")
    elif not re.match(r'[A-Za-z0-9]+', username):
      flash("name must contain only characters and numbers")
    else:
      insert_sql = "INSERT INTO users VALUES(?,?,?,?)"
      prep_stmt=ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, username)
      ibm_db.bind_param(prep_stmt, 2, email)
      ibm_db.bind_param(prep_stmt, 3, password)
      ibm_db.bind_param(prep_stmt, 4, ''.join(random.choices(string.ascii_letters, k=16)))
      ibm_db.execute(prep_stmt)
      flash("logged in")

    return redirect(url_for("login"))
  return render_template("reg.html")


@app.route("/login", methods=["GET", "POST"])
def login():
  if flask.request.method == "POST":

    data = flask.request.form
    username=data['username']
    password=data['pw']

    sql = "SELECT * FROM users WHERE username=? AND password=?"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.bind_param(stmt, 2, password)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    print(account)
    if account:
      session['LoggedIn'] = True
      session['userid'] = account['USERID']
      session['username'] = account['USERNAME']
      userid = account['USERID']
      flash("logged in")
      return redirect(url_for("dashboard"))
    else:
      flash("error")
  return render_template("login.html")



@app.route("/dashboard", methods=["GET", "POST"])
def dashboard():
  global request
  if flask.request.method == "POST" and session['LoggedIn']:
    if 'file' not in flask.request.files:
      flash('No file part')
      return redirect(flask.request.url)
    file = flask.request.files['file']
    if file.filename == '':
      flash('No image selected')
      return redirect(flask.request.url)
    if file and allowed_file(file.filename):
      filename = secure_filename(file.filename)
      file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
      flash('Image successfully uploaded')

      with open(os.path.join(app.config['UPLOAD_FOLDER'], filename), "rb") as f:
        file_bytes = f.read()

      request = service_pb2.PostModelOutputsRequest(
          model_id="food-item-v1-recognition",
          user_app_id=resources_pb2.UserAppIDSet(app_id=YOUR_APPLICATION_ID),
          inputs=[
            resources_pb2.Input(
              data=resources_pb2.Data(image=resources_pb2.Image(
                  base64=file_bytes
               )
              )
            )
          ],
      )
      response = stub.PostModelOutputs(request, metadata=metadata)

      if response.status.code != status_code_pb2.SUCCESS:
          print(response)
          raise Exception(f"Request failed, status code: {response.status}")

      foodname = response.outputs[0].data.concepts[0].name

      ingredients = ''
      for concept in response.outputs[0].data.concepts:
        ingredients += f"{concept.name}: {round(concept.value, 2)}, "

      nutritionValues = ''
      # nutritionApiUrl = "https://spoonacular-recipe-food-nutrition-
  v1.p.rapidapi.com/recipes/guessNutrition"
      # querystring = {"title":foodname}
      # headers = {
      #   "X-RapidAPI-Key": RAPIDAPI_KEY,
      #   "X-RapidAPI-Host": "spoonacular-recipe-food-nutrition-v1.p.rapidapi.com"
      # }
      # response = requests.request("GET", nutritionApiUrl, headers=headers, params=querystring)
      # nutritions = response.text




      url = "https://calorieninjas.p.rapidapi.com/v1/nutrition"
      querystring = {"query":foodname}
```

```
headers = {
  "X-RapidAPI-Key": "85887549f4msh51e7315b280a87ep1f43e0jsn585c940f2ea6",
  "X-RapidAPI-Host": "calorieninjas.p.rapidapi.com"
}
response = requests.request("GET", url, headers=headers, params=querystring)

nutritions = response.json()['items'];
nutritions = nutritions[0];
print(nutritions)

for i in nutritions:
  nutritionValues += f"{i}: {nutritions[i]}, "



sql = "INSERT INTO foods VALUES(?,?,?,?,?)"
stmt=ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, session['userid'])
ibm_db.bind_param(stmt, 2, datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
ibm_db.bind_param(stmt, 3, foodname)
ibm_db.bind_param(stmt, 4, ingredients)
ibm_db.bind_param(stmt, 5, nutritionValues)
ibm_db.execute(stmt)

# os.remove(os.path.join(app.config['UPLOAD_FOLDER'], filename))
return render_template("dashboard.html",
  filename = filename,
  username = session['username'],
  foodname = foodname,
  ingredients = ingredients,
  nutritionValues = nutritionValues,
)
```

```
   else:
     flash('Allowed image formats - png, jpg, jpeg')
     return redirect(flask.request.url)

  elif session['LoggedIn']:
   return render_template("dashboard.html", username=session['username'])
  else:
   return redirect(url_for("login"))



@app.route('/logout', methods=["GET", "POST"])
def logout():
  session.pop('LoggenIn', None)
  session.pop('userid', None)
  session.pop('username', None)
  return render_template("index.html")



@app.route('/display/<filename>', methods=["GET", "POST"])
def display(filename):
  print(filename)
  return redirect(url_for('static', filename='uploads/' + filename), code=301)



@app.route('/app', methods=["GET", "POST"])
def other():
  return render_template("index.html")



@app.route('/', methods=["GET", "POST"])
def index():
  return render_template("index.html")


if __name__ == "__main__":
  app.run(host ='0.0.0.0', port = 5000)
```